SPAR^{2}QL: From SPARQL to Rules*

Axel Polleres[1] and Roman Schindlauer[2]

[1] Universidad Rey Juan Carlos, Madrid, Spain
[2] Institut für Informationssysteme 184/3, Technische Universität Wien, Austria
axel@polleres.net, roman@kr.tuwien.ac.at

As the data and ontology layers of the Semantic Web stack have achieved a certain level of maturity in standard recommendations such as RDF and OWL, the current focus lies on two related aspects. On the one hand, the definition of a suitable query language for RDF, SPARQL, has just reached candidate recommendation status within the W3C. The establishment of the rules layer on top of the existing stack on the other hand marks the next step to be tackled, where especially languages with their roots in Logic Programming and Deductive Databases are receiving considerable attention. In this work we try to bridge the gap between these two efforts by providing translations between SPARQL and Datalog extended with negation and external built-in predicates. It appears that such a combination serves both as an underpinning for a more general rules and query language on top of RDF and SPARQL as well as for direct implementations of SPARQL on top of existing rules engines. Our prototype implementation is based on the datalog engine DLV. As it turns out, features of the language of this system can be fruitfully combined with SPARQL.

Motivation. SPARQL, currently under consideration as a candidate recommendation at W3C [5] provides a simple protocol and query language for RDF. Whereas syntactically SPARQL has apparent similarities with RDBMS query languages such as SQL, there are quite some semantic differences with respect to the underlying relational algebra, particularly related to blank nodes, as well as the UNION and OPTIONAL operators [4].

Analogies between Datalog and RDBMS query languages such as SQL are well-studied; fragments such as unions of conjunctive queries with nested subqueries involving set difference can be expressed by Datalog$^\neg$ (i.e., Datalog with negation) in a straightforward manner. Additionally, Datalog and its extensions by non-monotonic, unstratified negation and disjunction [1] offer additional expressive capabilities beyond these query languages along with a declarative semantics. However, no such investigations have yet been presented for SPARQL.

Existing engines for Datalog$^\neg$ such as DLV[3] or `smodels`[4] additionally handle aggregates [3], external predicates [2] such as import of RDF data, arithmetics,

* This work is partially supported by the Spanish MEC under the project TIC-2003-9001 and the Acción Integrada "Formal Techniques for Reasoning about Ontologies in E-Science", by the EC funded projects TripCom (FP6-027324) and REWERSE (IST-2003-506779), and by the Austrian Science Fund (FWF) grant P17212-N04.

[3] `http://www.dlvsystem.com`
[4] `http://www.tcs.hut.fi/Software/smodels/`

datatype-built-ins, etc. Remarkably, the underlying powerful rule language of these systems, often referred to as Answer Set Programming, is also considered as a basis for the Semantic Web rules layer in several works.

Translation and Prototype. By providing a translation from SPARQL into Datalog$^{\neg}$, we set the grounds for a smooth integration of powerful rules languages with the query layer on top of RDF. For instance, a SPARQL query

```
SELECT ?X ?Y
FROM <http://polleres.net/foaf.rdf>
WHERE { <http://polleres.net/foaf.rdf#me> foaf:knows ?X .
        OPTIONAL {?X foaf:mbox ?Y} }
```

can be straightforwardly expressed by the following rules (using the common PROLOG style notation):

```
triple(X,Y,Z) :- importRDF("http://polleres.net/foaf.rdf",X,Y,Z)
answer(X,Y) :- triple("#me","foaf:knows",X), triple(X,"foaf:mbox",Y).
answer(X,unbound) :- triple("#me","foaf:knows",X), not answer1(X).
answer1(X) :- triple(X,"foaf:mbox",Y).
```

Here, importRDF is an external predicate which allows the import of RDF triples from an external source. The extension of `answer` contains the pattern solutions of the query, where the constant `unbound` denotes unbound variables. Further external predicates are needed to cover, for instance, the evaluation of FILTER conditions, such as isBlank, isURI, etc. in SPARQL. We have developed an automatic translation covering all graph patterns in SPARQL which is fully compliant to the SPARQL semantics. A prototype has been implemented on top of the dlvhex system and is available at `http://con.fusion.at/dlvhex/sparql-query-evaluation.php` for evaluation. As it turns out, such a combination also allows for straight-forward extensions of features beyond the current version of SPARQL such as: Nested queries in ASK filters; transitive closure queries which seem essential in a graph query language such as SPARQL; as well as the addition of rules which can cover a larger subset of RDF(S)-entailment than simple RDF entailment adopted by the current SPARQL specification.

References

1. T. Eiter, G. Gottlob, H. Mannila. Disjunctive Datalog. *ACM Transactions on Database Systems*, 22(3):364–418, 1997.
2. T. Eiter, G. Ianni, R. Schindlauer, H. Tompits. Effective integration of declarative rules with external evaluations for Semantic Web reasoning. *ESWC 2006*, 2006.
3. W. Faber, N. Leone, G. Pfeifer. Recursive aggregates in disjunctive logic programs: Semantics and complexity. *JELIA 2004*, vol. 3229 *LNAI*. Springer, 2004.
4. J. Perez, M. Arenas, C. Gutierrez. Semantics and complexity of SPARQL. Technical Report DB/0605124, arXiv:cs, 2006.
5. E. Prud'hommeaux, A. Seaborne (eds.). SPARQL query language for RDF, 2006. W3C Candidate Recommendationi, available at `http://www.w3.org/TR/2006/CR-rdf-sparql-query-20060406/`.